Northwest University: ETL Project

Group 3: Andrew, Fiona, Liam

PROPOSAL

NBA Player Data

Our datasets include salary data of NBA players that played in the 2017-18 season. This set includes player name, team abbreviation, and salary. The other consists of NBA Players of the Week for both conferences from 1979-2020 with player name, team name (not abbreviated), conference, etc. We would join these two datasets and transform them so that they can be analyzed for player performance based on salary, using Player of the Week as a key metric.

DATASETS

https://www.kaggle.com/koki25ando/salary?select=NBA_season1718_salary.csv

https://www.kaggle.com/jacobbaruch/nba-player-of-the-week

Northwest University: ETL Project

Group 3: Andrew, Fiona, Liam

THE PROCESS

To begin, we extracted NBA Player of the Week ('POTW') and NBA Salary data from two datasets found on Kaggle:

- Player of the Week data
- NBA Salary Data

These datasets were both available in CSV format.

Next, we loaded these into a Jupyter Notebook and created dataframes for both POTW and Salary data. Because salary data was only available for 2017-2018, we used the .loc() function to isolate that season for both POTW and Salary data to trim the dataframe to just the 2017-18 NBA season.

We created a copy of the clean DF so that we could avoid destructive edits, if necessary.

We dropped columns from the POTW dataframe that weren't relevant for our analysis. The columns to keep are Player, Team, Position, and Season. We created a copy of the Salary dataset, again to ensure we are avoiding destructive edits. We then pared that dataset down to include columns 'Player','Tm','season17_18'.

We renamed the Tm and season17_18 columns to "Team Abv." And "Salary". In the POTW dataframe, we renamed the Team column to "Team Name". In the POTW set, we reset the index as it retained IDs from the original dataframe that was created prior to extracting only 2017-18 season data.

We had planned to reformat Salary numbers with commas so they would be easier to read, but this created issues in moving the dataframes to SQL. We left this commented out. We created new CSVs with only the 2017-18 season data that we were looking at.

THE PROCESS (cont.)

We tried to index on player name, but had issues turning the indexed DF into SQL, so we scrapped this step. At this point, the dataframes were ready to be imported into PGAdmin. We created a connection to our database (NBA2_db). An error we came across was our column names – they had to match with the SQL table column names, so we had to go back and rename the columns in our Pandas DFs. • (columns={"Player": "player", "Team Abv.": "team_abv", "Salary": "salary"}) • (columns={'Player':'player', 'Team Name':'team_name', 'Position':'position', 'Season':'season'})

We did engine.table_names() to ensure the SQL database was properly connected to our Jupyter Notebook. In our SQL database, we changed “id” to “index” after some errors getting the code “df.to_sql” to work. This worked and we were able to export both DFs into SQL.

We used PgAdmin, a relational database, for this project. The final tables to use are nba_players_of_week3 and nba_salary3 within the nba_schema.sql file.

We joined the tables on a RIGHT JOIN on player name in order to pull salary data from all POTW winners but exclude players in the 2017-18 season who had not received the POTW award. We Rejoiced!

Northwest University: ETL Project

Group 3: Andrew, Fiona, Liam

THE PROCESS

To begin, we extracted NBA Player of the Week ('POTW') and NBA Salary data from two datasets found on Kaggle:

- Player of the Week data
- NBA Salary Data

These datasets were both available in CSV format.

Next, we loaded these into a Jupyter Notebook and created dataframes for both POTW and Salary data. Because salary data was only available for 2017-2018, we used the .loc() function to isolate that season for both POTW and Salary data to trim the dataframe to just the 2017-18 NBA season.

We created a copy of the clean DF so that we could avoid destructive edits, if necessary.

We dropped columns from the POTW dataframe that weren't relevant for our analysis. The columns to keep are Player, Team, Position, and Season. We created a copy of the Salary dataset, again to ensure we are avoiding destructive edits. We then pared that dataset down to include columns 'Player','Tm','season17_18'.

We renamed the Tm and season17_18 columns to "Team Abv." And "Salary". In the POTW dataframe, we renamed the Team column to "Team Name". In the POTW set, we reset the index as it retained IDs from the original dataframe that was created prior to extracting only 2017-18 season data.

We had planned to reformat Salary numbers with commas so they would be easier to read, but this created issues in moving the dataframes to SQL. We left this commented out. We created new CSVs with only the 2017-18 season data that we were looking at.

THE PROCESS (cont.)

We tried to index on player name, but had issues turning the indexed DF into SQL, so we scrapped this step. At this point, the dataframes were ready to be imported into PGAdmin. We created a connection to our database (NBA2_db). An error we came across was our column names – they had to match with the SQL table column names, so we had to go back and rename the columns in our Pandas DFs. • (columns={"Player": "player", "Team Abv.": "team_abv", "Salary": "salary"}) • (columns={'Player':'player', 'Team Name':'team_name', 'Position':'position', 'Season':'season'})

We did engine.table_names() to ensure the SQL database was properly connected to our Jupyter Notebook. In our SQL database, we changed “id” to “index” after some errors getting the code “df.to_sql” to work. This worked and we were able to export both DFs into SQL.

We used PgAdmin, a relational database, for this project. The final tables to use are nba_players_of_week3 and nba_salary3 within the nba_schema.sql file.

We joined the tables on a RIGHT JOIN on player name in order to pull salary data from all POTW winners but exclude players in the 2017-18 season who had not received the POTW award. We Rejoiced!